DOCUMENT RESUME

ED 482 100 IR 058 788

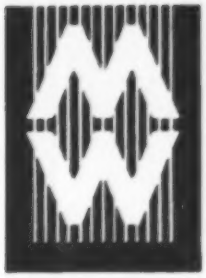
AUTHOR	Honeysett, Nik
TITLE	Content Management for a Content-Rich Website.
PUB DATE	2002-04-00
NOTE	12p.; In: Museums and the Web 2002: Selected Papers from an International Conference (6th, Boston, MA, April 17-20, 2002); see IR 058 778. Figures may contain very small and illegible font.
AVAILABLE FROM	For full text: http://www.archimuse.com/mw2002/papers/honeysett/honeysett.html/.
PUB TYPE	Reports - Descriptive (141) -- Speeches/Meeting Papers (150)
EDRS PRICE	EDRS Price MF01/PC01 Plus Postage.
DESCRIPTORS	*Computer System Design; Design Preferences; Material Development; *Museums; Web Sites; *World Wide Web
IDENTIFIERS	J Paul Getty Museum CA; Virtual Museums; *Web Site Design

ABSTRACT

Over the last year the J. Paul Getty Trust's Web presence has evolved from a group of disparate, independently maintained Web sites into a homogeneous consistently branded one. This transformation recently culminated with the implementation of a leading Content Management System (CMS). There were and are many process-changes and challenges in implementing a CMS in an institution such as the Getty. These issues are not unique and span the gamut from social, to business, to technical. This paper will highlight the major issues, describe the route the Getty took, and give an insight into the functionality and capability of a leading CMS application for a content-rich museum Web site. Topics covered include: the maintenance burden of re-launching the Web site before implementing a CMS; preparing for a CMS; CMS requirements; implementation; environment; templating; the templating process; workflow; deployment; the rollout; and future initiatives. (Author/ MES)

Reproductions supplied by EDRS are the best that can be made
from the original document.

ED 482 100



PAPERS
Museums and the Web 2002

Content Management For A Content-Rich Website

Nik Honeysett, J. Paul Getty Trust, Los Angeles, USA

http://www.getty.edu

PERMISSION TO REPRODUCE AND DISSEMINATE THIS MATERIAL HAS BEEN GRANTED BY

D. Bearman

TO THE EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)

1

U.S. DEPARTMENT OF EDUCATION
Office of Educational Research and Improvement
EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)

☑ This document has been reproduced as received from the person or organization originating it.

☐ Minor changes have been made to improve reproduction quality.

• Points of view or opinions stated in this document do not necessarily represent official OERI position or policy.

A&MI

Archives & Museum Informatics
2008 Murray Ave.
Suite D
Pittsburgh, PA
15217 USA
info@archimuse.com
www.archimuse.com

Abstract

Over the last year the J. Paul Getty Trust's Web presence has evolved from a group of disparate, independently maintained Web sites into a homogenous consistently branded one - getty.edu. This transformation recently culminated with the implementation of a leading Content Management System (CMS). There were and are many process-changes and challenges in implementing a CMS in an institution such as the Getty. These issues are not unique and span the gamut from social, to business, to technical. This paper will highlight the major issues, describe the route the Getty took, and give an insight into the functionality and capability of a leading CMS application for a content-rich museum Web site.

Keywords: Content Management, Templates, Workflow, Virtualisation, XML.

Introduction

The Getty is a campus of six programs: Museum, Research Institute, Conservation Institute, Trust Administration, Grant Program and Leadership Institute. All these programs contribute content for the Web site, resulting in a very content-rich environment. It currently includes details on 4,000 works of art; 1,500 artist biographies; 3 hours of streaming video; past and present exhibitions; art historical research papers and tools; conservation research papers and tools; on-line library research tools; detailed visitor information; an event calendar, and a parking reservation system.

Up until a year ago, each program independently maintained its own portion of getty.edu. A gateway homepage was placed at the top level, but as soon as one drilled down into the Website, it was immediately apparent that there was no consistent design, navigation, or look and feel, and no way to navigate around the site without going back to the homepage.

A trustee-level decision was made to present the Web site with a consistent design and look and feel - a Web group was formed and charged with this task. Accomplishing this task would include the implementation of a Content Management System (CMS) to ensure that all the programs could continue to contribute content to the Web site but in a managed and decentralized way. Rather than implement the redesign and CMS simultaneously, the redesign was implemented first, but significant effort was assigned to preparing the HTML pages for later CMS integration.

IR058788

BEST COPY AVAILABLE

The Web site was re-launched in February, 2001, with a unified design and functionality and a more thematic treatment of the content. The task had required that the webgroup temporarily 'own' all the content and the processes associated with getting that content to the Web. The knowledge gained during this process was crucial in being able to identify the real requirements of any CMS that the Getty would need to implement.

Maintenance Burden

There was a significant burden to re-launching the Web site before implementing a CMS. To guarantee that ongoing development and maintenance on the Web site adhered to the new rules and style guide, all content still had to funnel through the webgroup - the new design had committed us to a frequent subsite update regime. The GartnerGroup, whom the Getty use to benchmark themselves against similar organizations and business processes, publish a graph which exactly summed up where we stood in the Web life-cycle. Even with a webgroup of 18 people, there were few resources to develop new content.

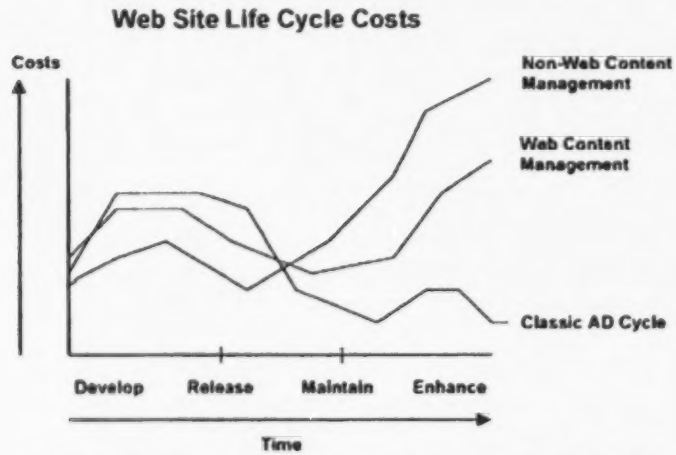


Fig 1: Website Life Cycle Costs. GartnerGroup © 1999

Preparing For a CMS

To prepare the Getty community for a CMS, which might take 6-9 months to select, the webgroup temporarily established its own managed environment to create and deploy content to coincide with the launch of the redesign. On the infrastructure side, we implemented a three-tier staging environment (see figure 2) and a custom-written application to 'push' content between tiers. This was a fairly high-level tool operating on directories rather than individual files. On the business side, we established a strict push schedule, which meant that content could only be moved into production twice a week - although this could be fast-tracked for emergencies on a case by case basis.

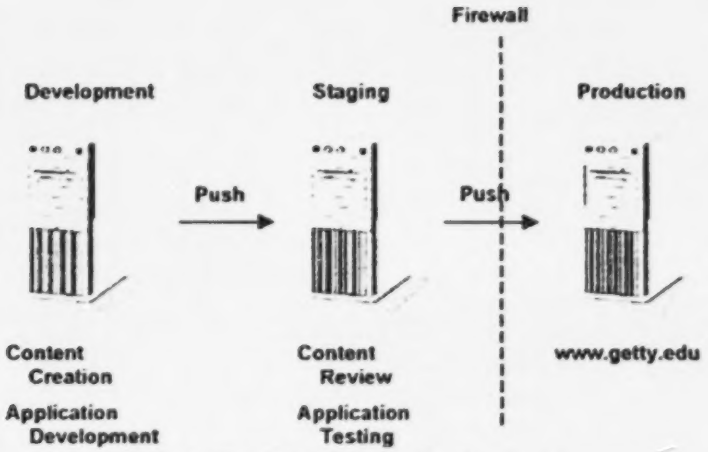


Fig 2: Three-tier Staging Environment

Initially there was great resistance from the programs to this regime. The 'anarchic' nature of content generation and deployment that had established itself at the Getty meant that this was a considerable culture shock. The programs perceived a 'loss of control' of their content, and objected to the delay between pushes. Fortunately, the reality was different: the schedule of pushing every other day allowed for only one day to create and review content before it went live. The programs soon fell into the regime and altered their business processes to accommodate the schedule. With the CMS implementation we could afford to increase the push schedule to whatever was appropriate - since it would be automatic. But the temporary situation was a good proving ground for a professionally managed environment.

CMS Requirements

Armed with the detailed knowledge of content generation and processes around the Getty gained during the redesign, the Web group drew up a requirements document for a CMS. Our 'big ticket' items are probably consistent with many museums' CMS wish list, and these were:

- Template Driven: To guarantee consistent visual design, to separate content from design and to allow multiple content use.
- Used by HTML Illiterate: To allow an interface that lets non-HTML contributors submit and review web content.
- Workflow: The processes established at the Getty to get content to the Web site range from a simple two-step workflow to complex multi-program, concurrent-task processes.
- File Versioning: The ability to track individual file changes, what and by whom.
- Rollback: The ability to publish an edition of the Web site on any previous date.
- Open/Standard Architecture: The Getty's Web infrastructure is based on Unix servers, iPlanet Web servers, Java/perl CGI applications and Oracle databases, all sitting on a Novell network. Any Web application has to live in this environment.
- Virtualisation: The ability to see any changes or new content in the context of the entire Web site - before it is deployed.
- Multiple Web site Management: For getty.edu and for our Intranet GO (Getty Online).

- Cross-platform clients: The Getty has Mac and PC-based contributors.
- Multi-channel Deployment: Eg. XML and WML delivery.
- In-house Maintenance and Development: Not to have to go back to the vendor.
- Vendor Stability: An important consideration given the current economic times.

After tracking the CMS market during the re-design, the Web group focused on three CMS products after the usual RFI (Request For Information) and RFP (Request For Proposal) processes. These were: Stellent's Expedio (http://www.stellent.com), Vignette's StoryServer (http://www.vignette.com) and Interwoven's Teamsite (http://www.interwoven.com). After a bake-off, the Getty selected Interwoven as its preferred CMS vendor.

Implementation

Teamsite is a Web-application suite based on C++, java, perl/cgi and XML technologies. It is a client/server environment with all users interacting with it via a browser. After some testing we established that Internet Explorer was more compatible. The only exception to the browser interface is the module to create workflows – this is a Windows application. Teamsite offers fully integrated **Templating**, a very flexible and fully integrated **Workflow** engine, full **Virtualisation** and integration with a broad range of Web content creation applications such as **Dreamweaver** and **Homesite**. It also stands as a solid platform on which to further develop our Web site, by integrating with an array of **application**, **personalization** and **syndication** servers – which are long-term goals for the Web site.

Interwoven's terms and conditions of sale include an agreed method of installation and configuration. This means you need to buy their professional services or contract with a registered Interwoven partner - a so- called 'enabler', of which there are many and for which there is a 'corkage' fee to Interwoven. After reading a number of 'dot com' elegies, we were keen not to have any one of these 'enablers' come in, install the product, then hold us to ransom over professional services to develop our site. Also, our Web group has a strong technical component and should be more than capable of maintaining the environment. The best integration solution for us was Interwoven's so-called 'Fast Forward Core Pack' - a 30-day engagement with a Teamsite consultant and project manager. At the end of this engagement, the client is 'guaranteed' to have a correctly installed and configured product, at least one site deployed through Teamsite as a file-managed process, and up to two templates and two workflows in place.

We scheduled technical training to coincide with the arrival of the implementation consultant so that we could speak the Teamsite language and could assist in the installation and configuration. We installed the suite onto our Unix **staging** server, since once operational, we eliminated the need for a separate staging environment.

Environment

Teamsite uses a server's native file structure to manage and deploy content, and manages separate Web sites as **branches**. Because of this

approach, the default view of a Web site within Teamsite is as a file system like Windows Explorer:

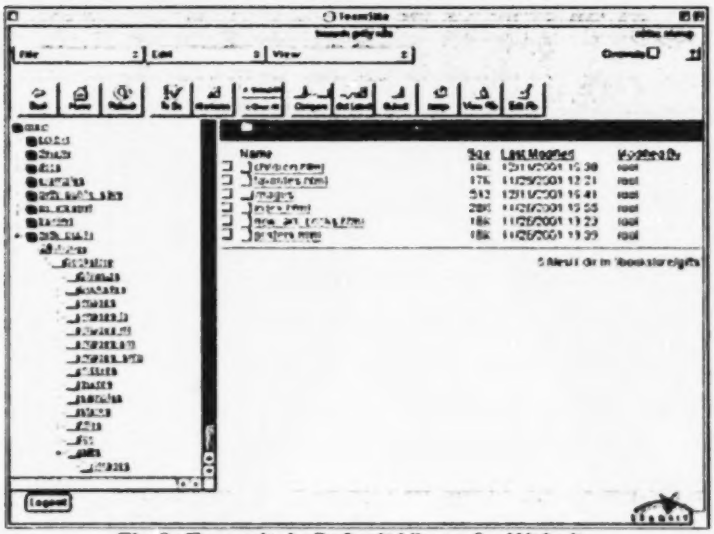


Fig 3: Teamsite's Default View of a Website

Each registered user has an account based on the server's native authorization technology. In our case this is LDAP. Within Teamsite, each user is assigned at least one **workarea** and has a virtual view of the whole site. This **workarea** corresponds to the section of the Web site on which they work; they only have permission to modify documents within their own **workarea**. Every user is also assigned a **role** depending on contribution level. This determines the authority they have within the environment: **author**, **editor**, **administrator** or **master**:

- Author – the primary content creator: he owns content; can create and edit files; can receive assignments through the workflow engine and has work approved by his workarea owner.
- Editor – creates content and oversees content creation: he owns a workarea; can create, edit and delete files; approves or rejects work of the authors; can submit files to the staging area and can publish editions.
- Administrator – manages an area or branch (Web site), of which he is the owner and can create and delete workareas.
- Master – has absolute power and fundamental administrative control over the product - a role reserved for a Unix System Administrator and a person to be chosen wisely.

BEST COPY AVAILABLE

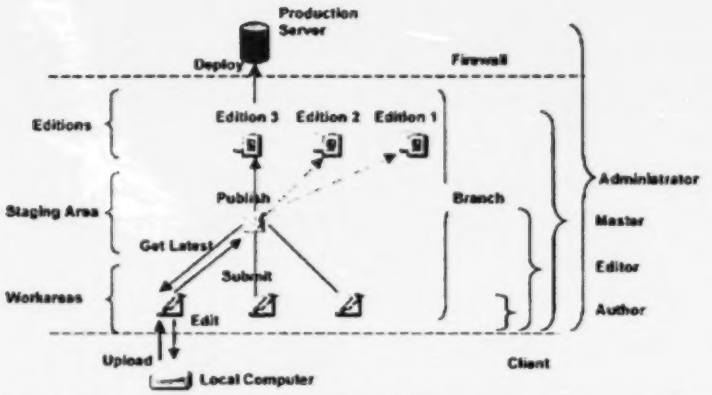


Fig 4: Schematic of the Teamsite Environment and Associated Roles

Working within the **workarea**, a user first synchronises with the latest version of content by performing a **get latest** function within the **staging area**. When the appropriate edits have been made and checked within the virtualized view of the entire site, the content is submitted back to the **staging area**. Browsing in the virtualized view of the Web site is no different from browsing the site under normal circumstances. The **staging area** is where all the content from different **workareas** is integrated and tested. It is a read-only environment where nothing can be edited. According to the publishing schedule, a snapshot is made of the **staging area** creating an **edition**, which can then be deployed to the production server.

We were looking to manage two Web sites with Teamsite: getty.edu and our Intranet, GO. After a week of analysis we began planning two development environments, each of these becoming a branch within Teamsite. We then 'sucked in' each site – a tar'ing, FTP and untar'ing process – which took about a day each. So, within two week' of the consultant's arrival, we had set up a pre-production environment for both sites. With these environments in place, we scheduled some intensive training on the core Teamsite functions of Templating and Workflow – unfortunately, this required a lengthy field trip to Interwoven's training facilities in San Francisco.

Templating

Templating is a core function of a CMS. It allows the separation of content from design, the automatic generation of an HTML page from content, guarantees consistent visual design, and allows Web pages to be generated by HTML-illiterate contributors. A fully templated site is one significant goal of full implementation. Teamsite's templating system is based on XML and requires a thorough understanding to develop templates. We began by analyzing our Web site to identify a list of initial template candidates. We started with the broad assumption on the cost/benefit of converting pages under the homepage. This indicated that the greatest cost/benefit was to be had at the lowest levels, where many pages would be served with a single template. For example, the museum collection subsite consists of approximately 18,000 pages,. Analysis of the design and layout indicated that we could probably serve this with fewer than five templates, possibly even one.

BEST COPY AVAILABLE

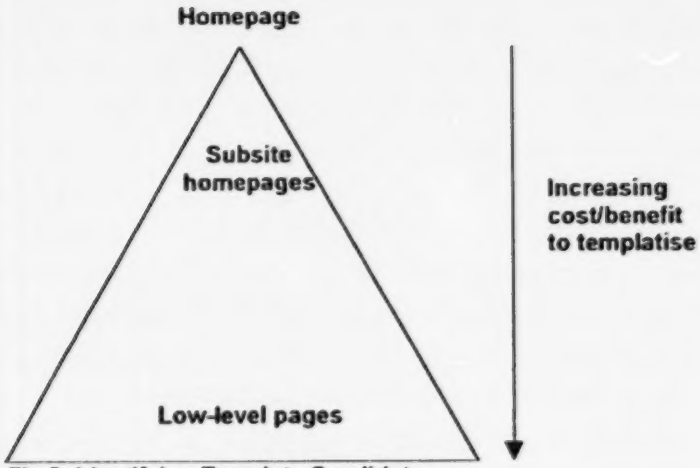


Fig 5: Identifying Template Candidates

We then applied a number of qualifications: Do a number of people contribute content to this page? Does the content on this page change often? Additionally, we wanted the first template candidates to also establish precedents for resolving issues that would globally impact the site. We focused on two candidates: the Job Postings, which have a lot of similar pages and a daily turnover, and the News Articles, which have a weekly turnover but a significant number of contributors (and consequently a complex associated workflow).

Templating Process

Templating is a three-step process of data capture, storage and presentation, figure 6 shows a schematic of this process.

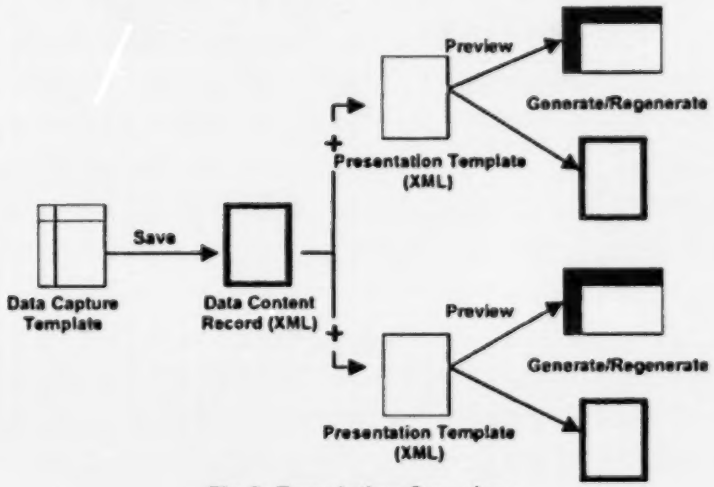


Fig 6: Templating Overview

When a user wants to submit content using a template, they first fill out a **data capture template** (**DCT**). This is in an XML configuration file, created by a developer with a good understanding of XML and some

basic programming skills. Submitting a completed **DCT** saves a **data content record** (**DCR**) which is either saved as a file or to a database. The last element is a **presentation template** (**PT**), another XML file with a variety of markup flavors: ASP, JSP, WML or HTML and also embedded **perl** callouts and conditional programming tags. The **PT** is essentially created from an existing HTML page by giving it an XML 'wrapper' and substituting the content areas with appropriate XML tags that are defined in the **DCR**. Figure 7 shows code fragments tracing a single piece of content, a job title, from capture to presentation:

DCT fragment
Capturing a 'job title' field

```
<?xml version="1.0" sta
<!DOCTYPE templating
-
<item name="jobtitle">
  <text maxlength=40
        required=true
        size=40 />
</item>
```

DCR fragment
Storing the 'job title' field

```
<?xml version="1.0" sta
<!DOCTYPE record
-
<item delimiter=", "
      name="jobtitle"
      <value>
        Web Manager
      </value>
</item>
```

PT fragment
Capturing a job title field

```
<?xml version="1.0" sta
-
<TR>
  <TD><font size="+2"
    <iw_value
    name='dcr.jobtitle'
    </iw_value>
  </TD>
<TR>
```

Fig 7: Tracing a single piece of content

When the presentation engine combines the **DCR** and the **PT** it executes any perl and any conditional tags to generate the final page. The jurisdiction of the different roles in the templating process can be summarized as:

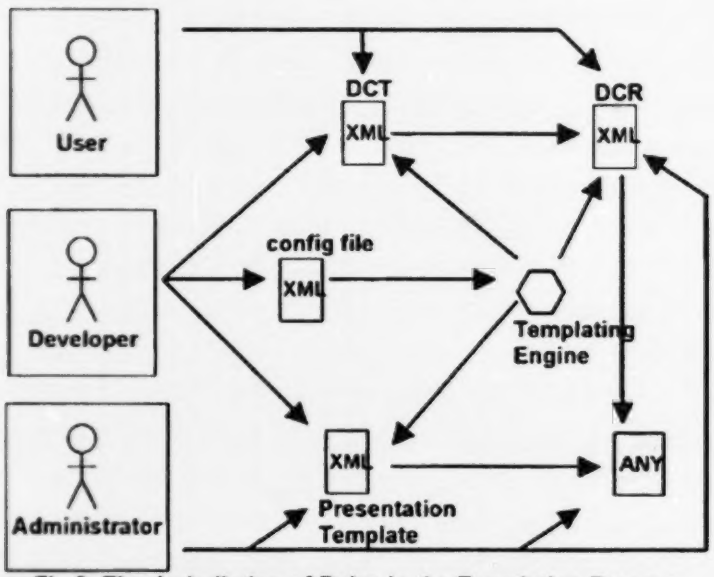


Fig 8: The Jurisdiction of Roles in the Templating Process

The templating architecture makes for a very powerful and flexible process and allows for single presentation templates to account for a wide range of related Web pages. Much of the template process can be automated; for example, workflows can be invoked at the **DCR** commit stage, or the final Web page can be generated automatically when a **DCR** is saved. It is wise not to underestimate the amount of work required to convert a Web site into templates.

BEST COPY AVAILABLE

Workflow

Workflow is another core function of a CMS - it defines the content creation and approval processes. The workflow module in Teamsite was one of the most comprehensive and flexible we reviewed during the selection process. The biggest challenge we found in implementing workflows within the Getty is the analysis of the 'business' processes, which continues to be a challenge. This analysis is a time-consuming practice in an environment where these processes have grown organically over time. When interviewing staff around the Getty, we often found ourselves giving advice on consolidating their processes, since no one had thought to review what was happening. Often when we flowcharted the program's processes, staff were surprised at how redundant their activities were. The concise steps to analyzing a process that we use are:

1. State the general requirement for the process

2. Identify the tasks that need to be performed

3. Identify who needs to perform the tasks

4. Order the tasks in which they need to be performed

5. Review with the process requirement

Having generated and signed off a flow chart representing the workflow process, the next step is to implement the workflow. **WorkflowBuilder** is a drag-and-drop flowcharting application based heavily on the **Visio** interface.

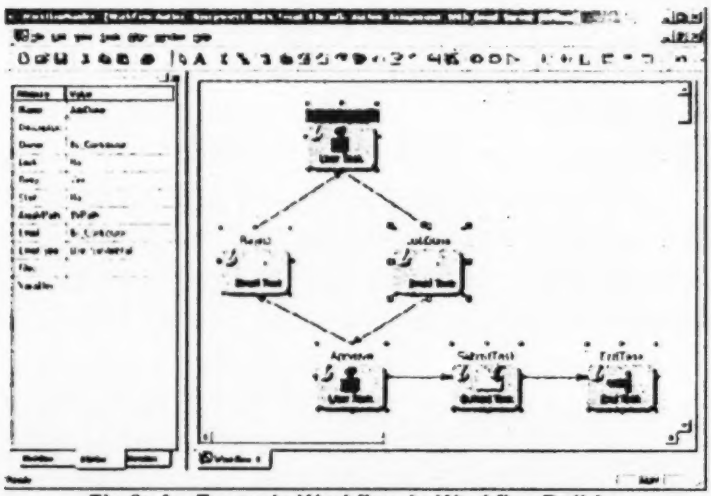


Fig 9: An Example Workflow in WorkflowBuilder

Figure 9 shows a typical workflow of six **tasks**. When invoked within Teamsite, the instance of this workflow is termed a **job**. It has a variety of variables associated with it, which are defined when the **job** starts. For example, one variable might be the name of a file that needs updating or the name of a directory where a new page needs to be created. The first task in figure 9 is a content-creation task which will be assigned to a user

BEST COPY AVAILABLE

using the **owner** variable. When the content has been written and submitted, the workflow sends an e-mail to the appropriate **approver** with review instructions and a hyperlink to the relevant content. A rejection generates an e-mail back to the **owner**, but an approval submits the content which might invoke the automatic generation of a Web page ready for deployment. This workflow is very generic. The use of variables allows for many instances of a single workflow – the variables simply travel along with a particular workflow instance. The power of the workflow engine can be extended immensely by additional tasks such as the **CGI** script invocation and an **external** task which runs any external application or script on the host machine.

Deployment

Deployment of content is the final step in the content generation process. It is the sending and receiving of files from the base server to the receiver host. Any number of deployment scenarios can be created, based on three basic themes: Deploy files in a specified list - the list can be generated programmatically; Deploy files in a directory by file comparison between the source and the target; Deploy directories by directory comparison between the source and the target. The file transfer executes in a transactional mode, meaning that content will not be replaced on the production server until all the requested files have been received.

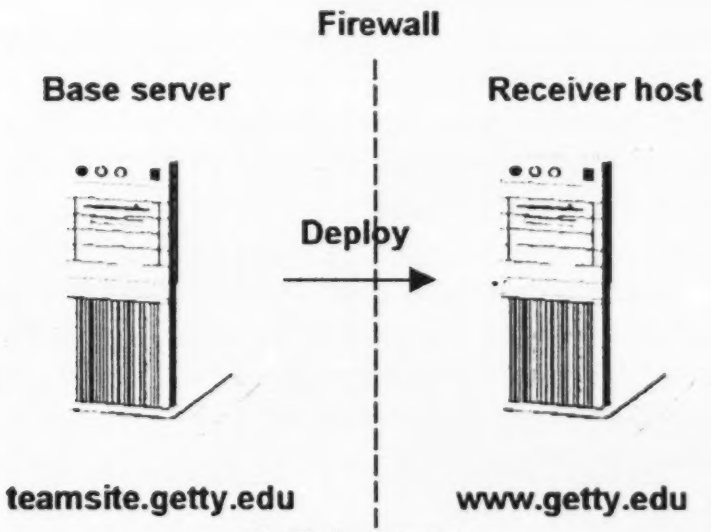


Fig 10: Deployment

The Rollout

One important factor in selecting this product was its ability to manage a Web site at a file level. This allowed us to implement Teamsite without disturbing the contributors around the Getty using their Dreamweaver and Homesite applications. We simply continued to import their files from the development server into Teamsite until we were ready to schedule their training and register them as users. The go-live date was an uneventful day. One day we were 'pushing,' and the next day we were 'deploying'.– This fact alone makes it a resounding success. The process of trawling through the Website leaving templated content in our

wake is ongoing, to be completed around June 2002.

Future Initiatives

We selected Teamsite with a long-term web strategy in mind. We chose this particular product because we require a platform that will foster the best opportunities to meet our long term goals and also one that is flexible enough to integrate with presently unknown program initiatives and future www trends. There are two immediate initiatives that are of note:

Vocabulary-Assisted Metatagging – There is a significant metadata initiative at the Getty and extensive research into vocabulary-assisted searching. Currently, searching on getty.edu results in an 'invisible' expansion against our **ULAN** (United List of Artist Names) vocabulary. By way of an example, if one searches for Carrucci on our Web site, 15 hits are returned to Pontormo but no mention of Carrucci – because Carrucci was the birth name of Pontormo as defined in ULAN. While we will continue to expand search queries against our vocabularies (**Art & Architecture Thesaurus** and the **Thesaurus of Geographic Names** are planned shortly), we plan to integrate more fully the metatagging of content at the creation stage. As luck would have it, Teamsite has a module called **Metatagger**, which we have already begun to investigate.

Personalisation – The 'next big thing' for our Web site is a venture into the world of Personalization. Again, as luck would have it, Interwoven is partnered with a variety of personalization server providers. At the time of writing, we are in the selection process, so stay tuned.

Conclusion

Teamsite is an expensive solution to enterprise-wide content management that the Getty is in the fortunate position to be able to afford. Moreover, our Web strategy is such that we require an 'industrial grade' CMS such as this to achieve our goals. The implementation involved a significant amount of planning, strategizing and work plus a high level of technical skill and support. The ongoing maintenance and development of the site through Teamsite has greatly increased the skill level of the Web group and shifted the typical requirements of a 'manually' maintained Website. In some areas it has polarized the webgroup resources, requiring more technical people to work on the 'back end' but fewer technical and fewer Web-literate people on the 'front end'. The executive-level expectation of a CMS is often a reduction in resources required for Web site development. To a certain extent this is true; however, those reduced resources are at a higher technical level that may offset any perceived budgetary reductions. For us, the webgroup is just as busy after the implementation. The difference is that we're accomplishing more.

U.S. Department of Education
Office of Educational Research and Improvement (OERI)
National Library of Education (NLE)
Educational Resources Information Center (ERIC)

NOTICE

Reproduction Basis

[X] This document is covered by a signed "Reproduction Release (Blanket)" form (on file within the ERIC system), encompassing all or classes of documents from its source organization and, therefore, does not require a "Specific Document" Release form.

[ ] This document is Federally-funded, or carries its own permission to reproduce, or is otherwise in the public domain and, therefore, may be reproduced by ERIC without a signed Reproduction Release form (either "Specific Document" or "Blanket").